Developing and Using Audio Units

Session 203

These are confidential sessions—please refrain from filming or taking pictures.

Graphics and Media

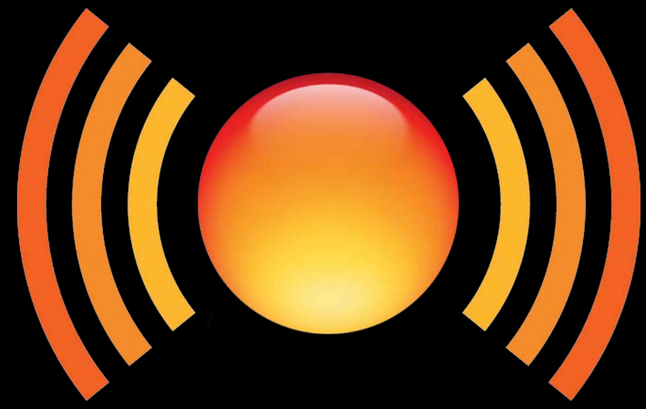
Developing and Using Audio Units

Chris Rogers and Michael Hopkins
Core Audio Engineering

http://developer.apple.com/wwdc2005/

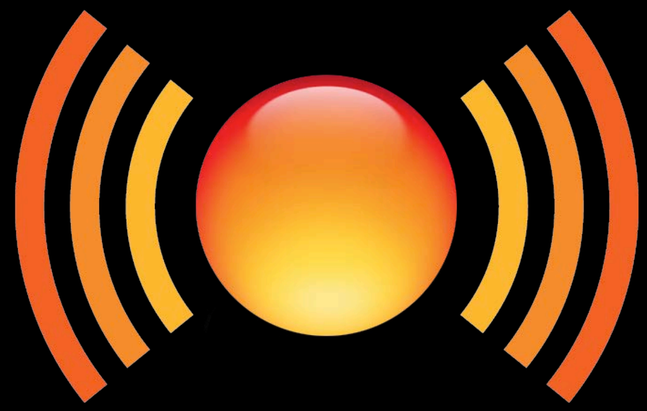
What Is an Audio Unit?

- Plug-in specification for audio
- Packaged as a Component
- Host uses the Component Manager to find and open an instance



Why Write an Audio Unit?

- System-wide
- Hosted by top audio applications
  - Apple audio applications (Logic, GarageBand, …)
  - Digital Performer
  - Ableton Live
  - Many others…
- Adds value to your audio technology



Audio Unit Types

- Effects
- Instruments
- Music Effects
- Generators
- Mixers
- Output Units
- Format Converters
- Offline Units

Audio Unit Examples

- Effects and Music Effects
  - Reverb
  - Graphic EQ
  - Pitch Shifter
- Instruments
  - Sample playback synth
  - Analog modeling synth
  - Physical modeling synth

Audio Unit Examples

- Generators (new in Tiger)
  - AUFilePlayer
  - AUNetReceive
- Mixers
  - AUMatrixMixer
  - AUMixer3D

Audio Unit Examples

- Output Units
  - Audio Device Output
- Format Converters
  - Sample-rate conversion
  - AUTimePitch
- Offline
  - Normalizer
  - Reverse

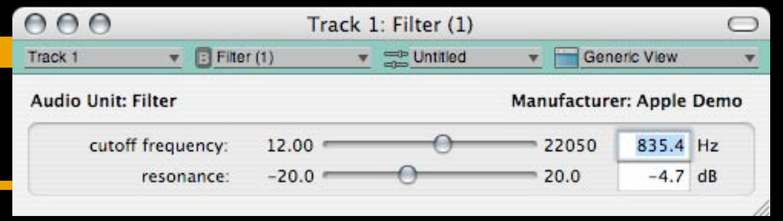
Demo

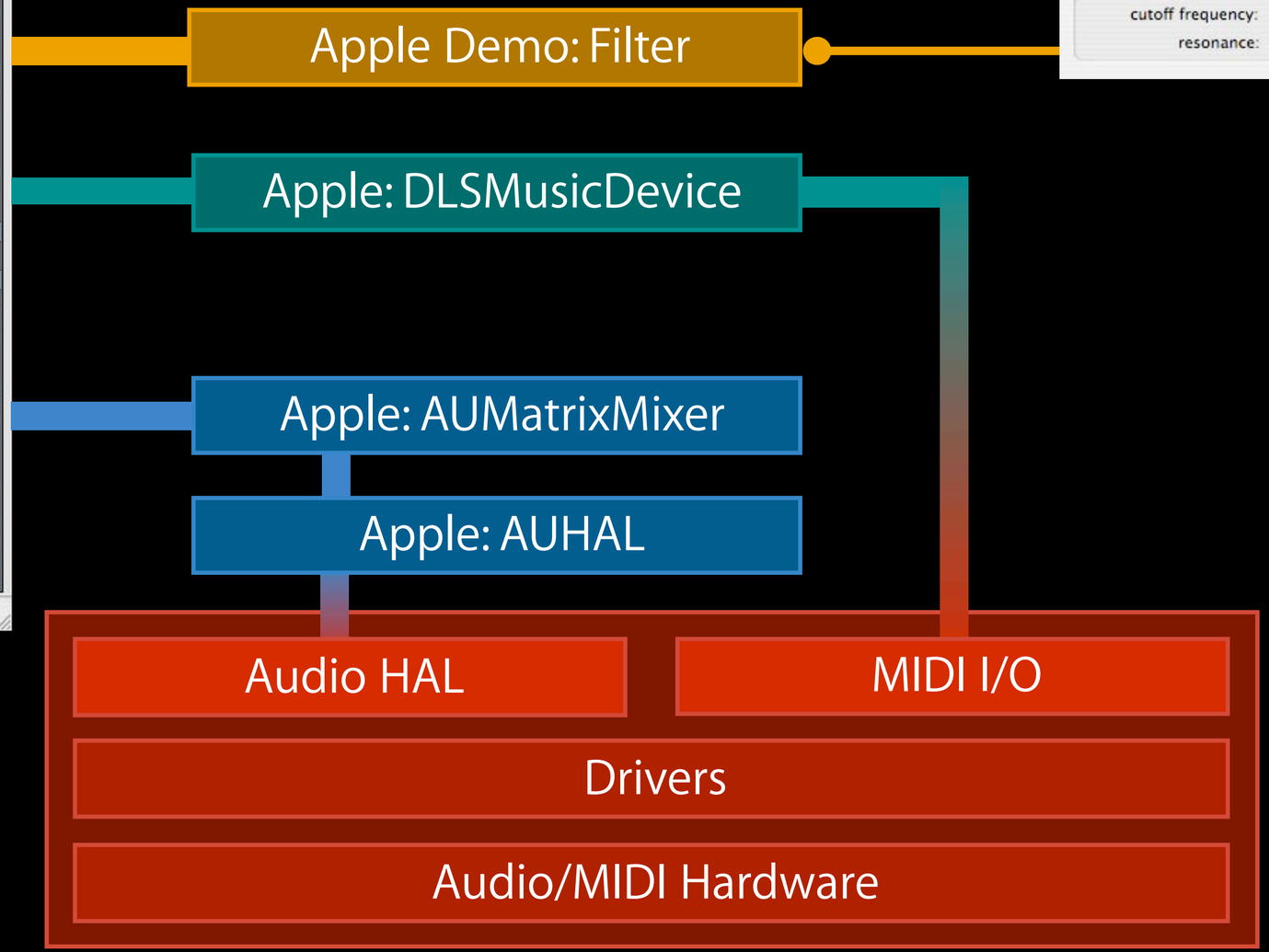
Running an Audio Unit in AU Lab

Chris Rogers
Software Engineer

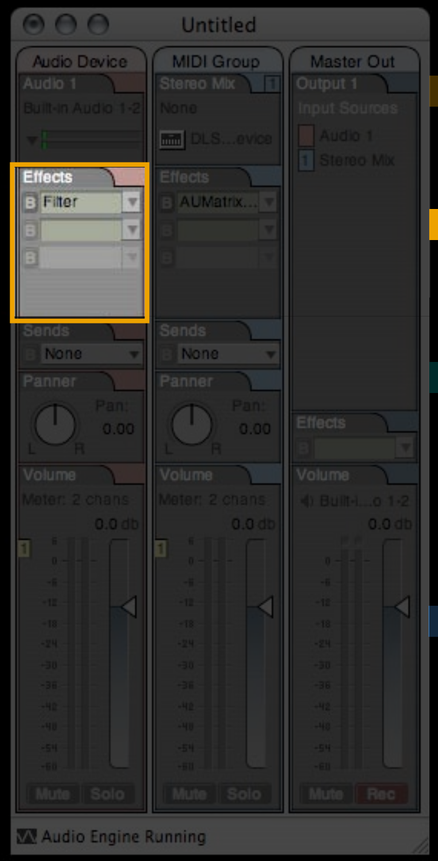
http://developer.apple.com/wwdc2005/

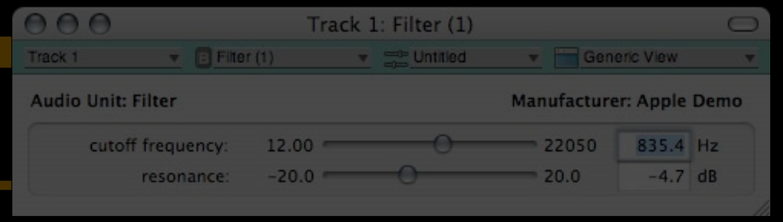
Audio Unit Host Application

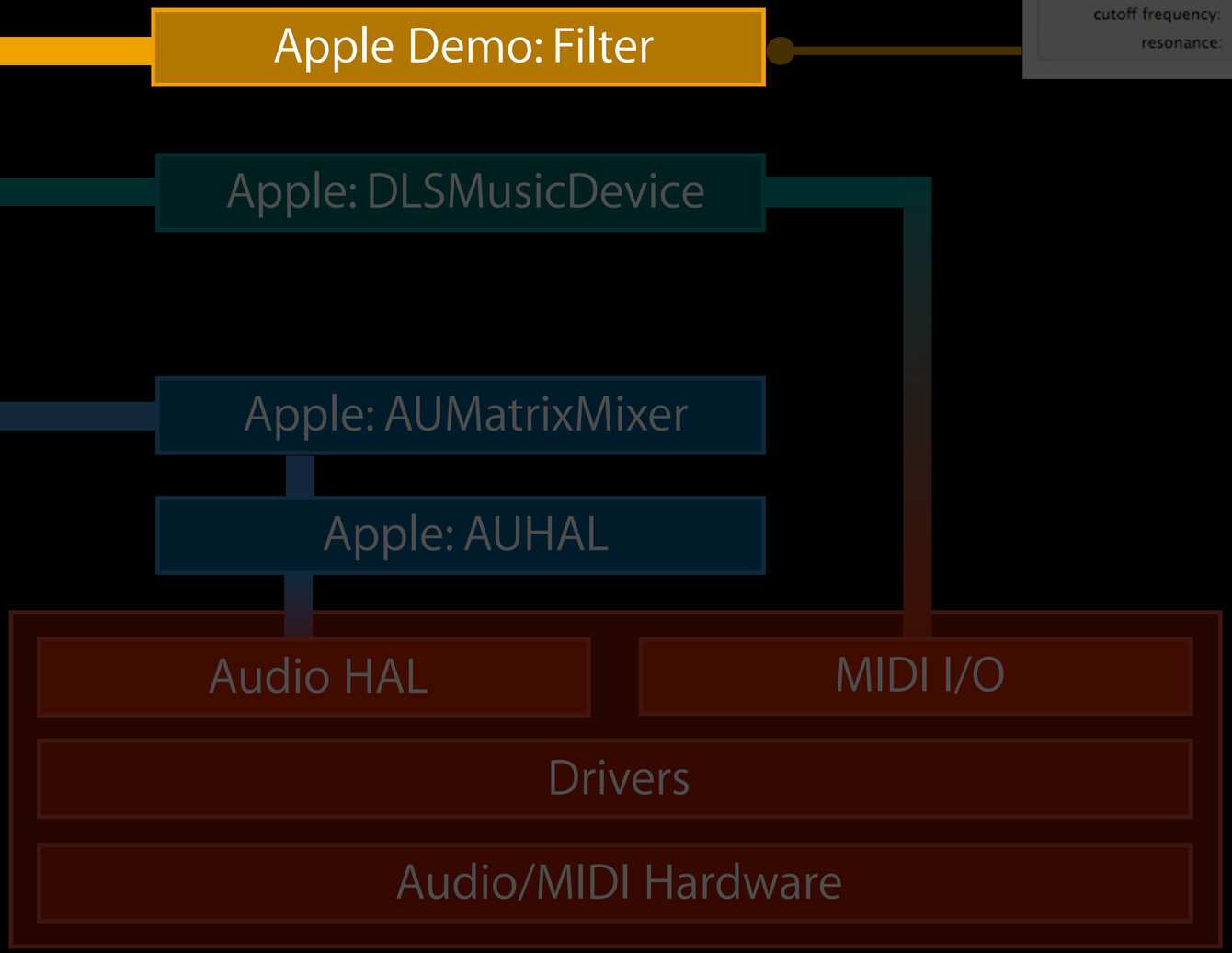




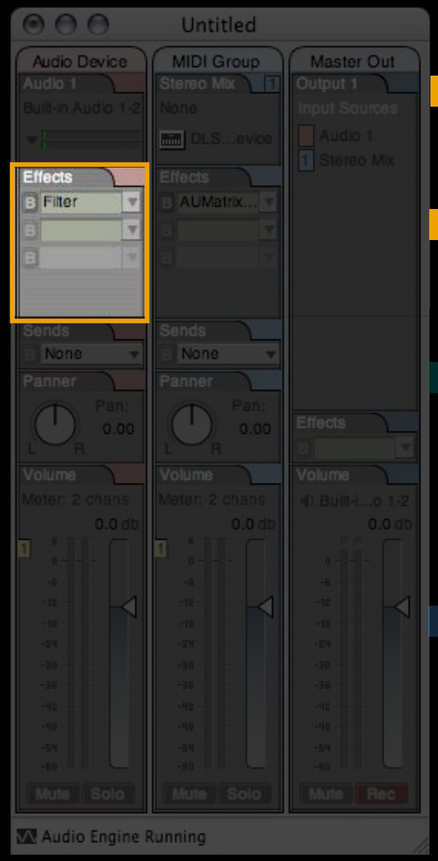
Audio Unit Host Application

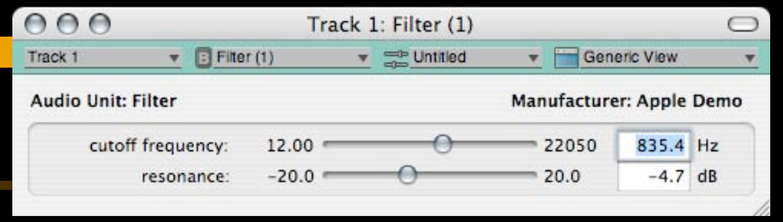






Audio Unit Host Application





Apple: Filter

Apple: DLSMusicDevice

Apple: AUMatrixMixer

Apple: AUHAL

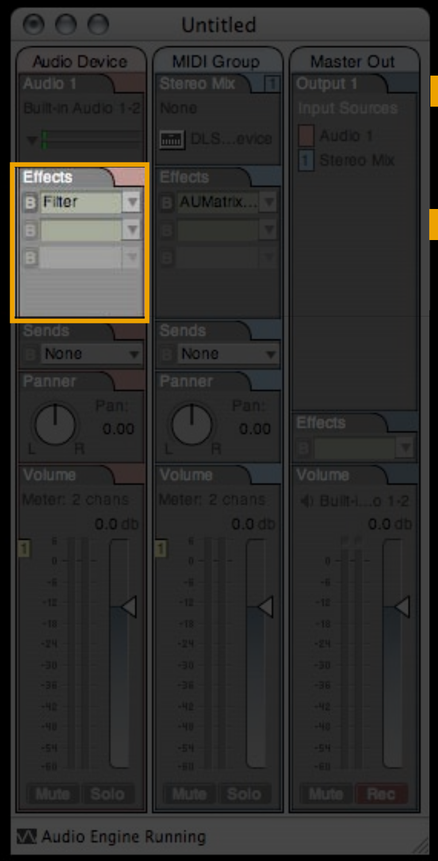
Audio HAL

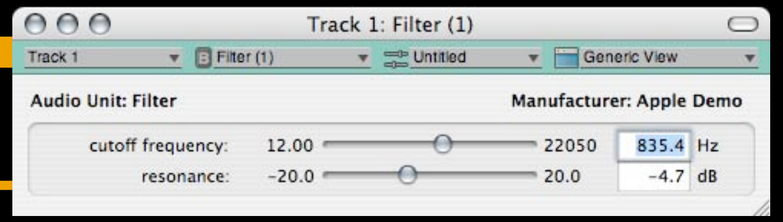
MIDI I/O

Drivers

Audio/MIDI Hardware

Audio Unit Host Application





Apple: Filter

Apple: DLSMusicDevice

Apple: AUMatrixMixer

Apple: AUHAL

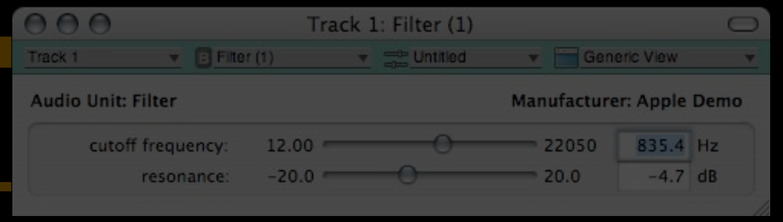
Audio HAL

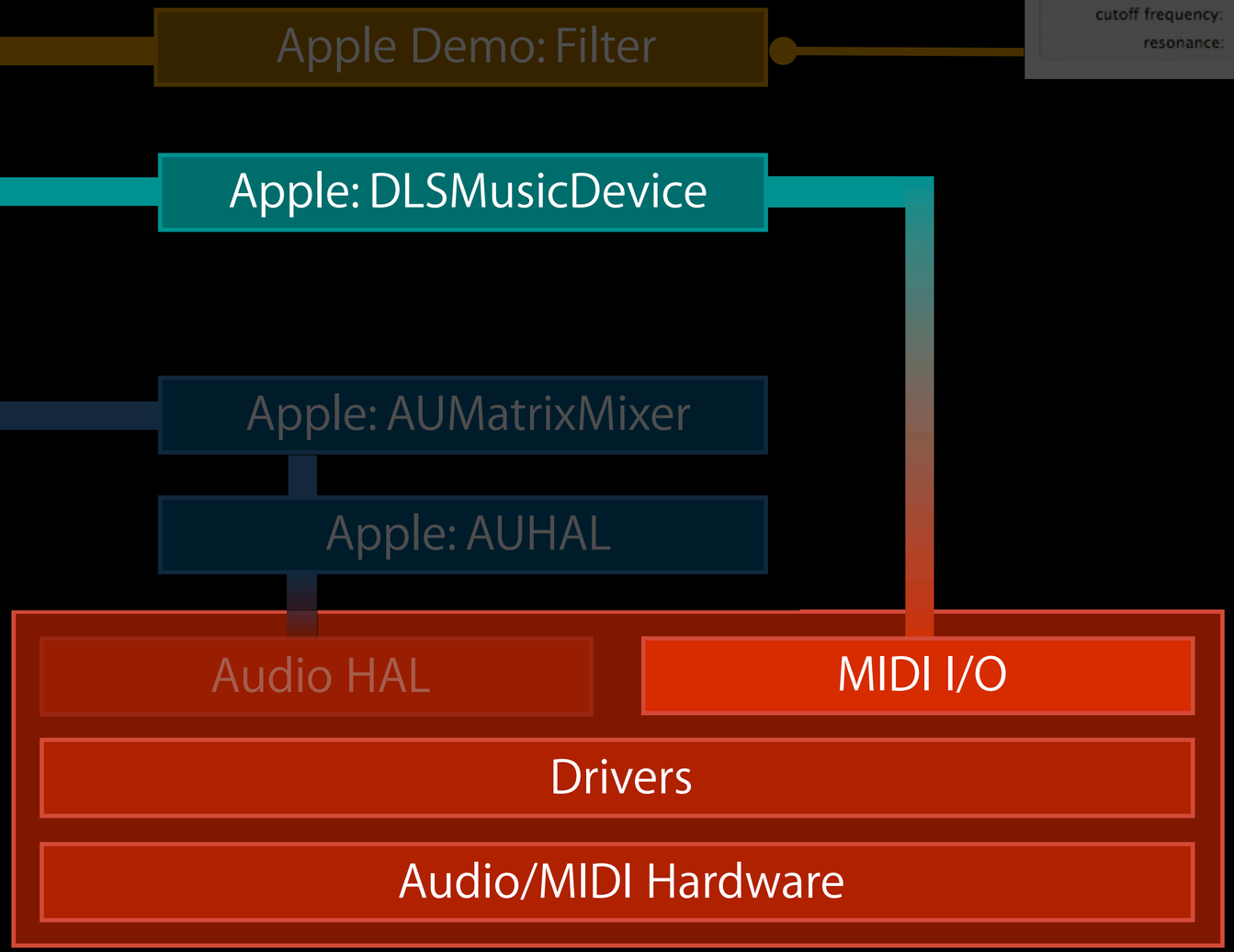
MIDI I/O

Drivers

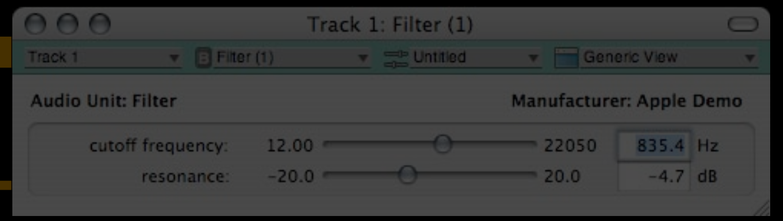
Audio/MIDI Hardware

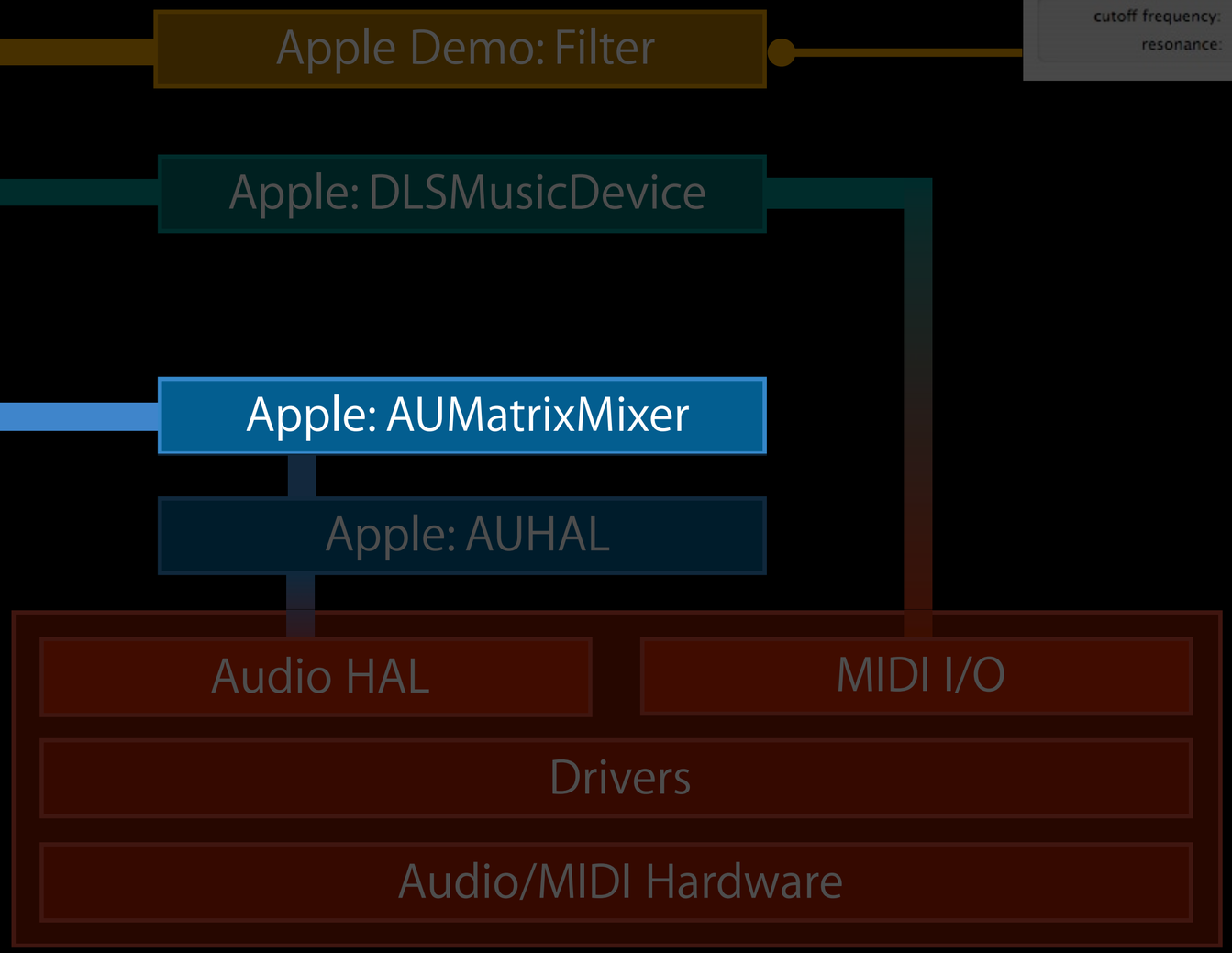
Audio Unit Host Application



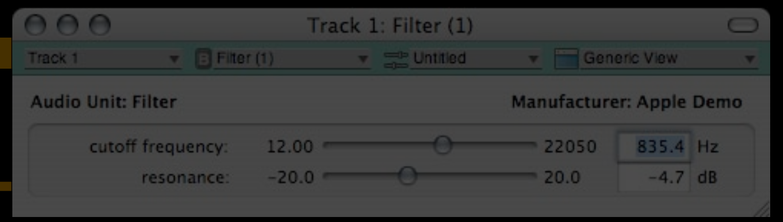


Audio Unit Host Application





Audio Unit Host Application



Apple Demo: Filter

Apple: DLSMusicDevice

Apple: AUMatrixMixer

Apple: AUHAL

Audio HAL

MIDI I/O

Drivers

Audio/MIDI Hardware

Directory Structure

The anatomy of an Audio Unit with a Custom Cocoa UI

- Audio Unit is contained in a .component bundle

FilterDemo.component	
Info.plist	(Bundle ID, version number, …)
MacOS	
FilterDemo	(Audio Unit executable code)
Resources	
CocoaFilterView.bundle	(Cocoa view bundle)
English.lproj	(AU Localized resources)
FilterDemo.rsrc	(Component description: type, subtype, manufacturer, etc.)

Installing the Audio Unit

Copy FilterDemo.component to

~/Library/Audio/Plug-Ins/Components/

or

/Library/Audio/Plug-Ins/Components/

FilterDemo.component

Validating the Audio Unit

With the auval command-line tool

- Checks that Audio Unit will work robustly in host applications
- Creates a more stable experience for your users
- Some hosts **require validation!**

Creating a Filter Effect Audio Unit

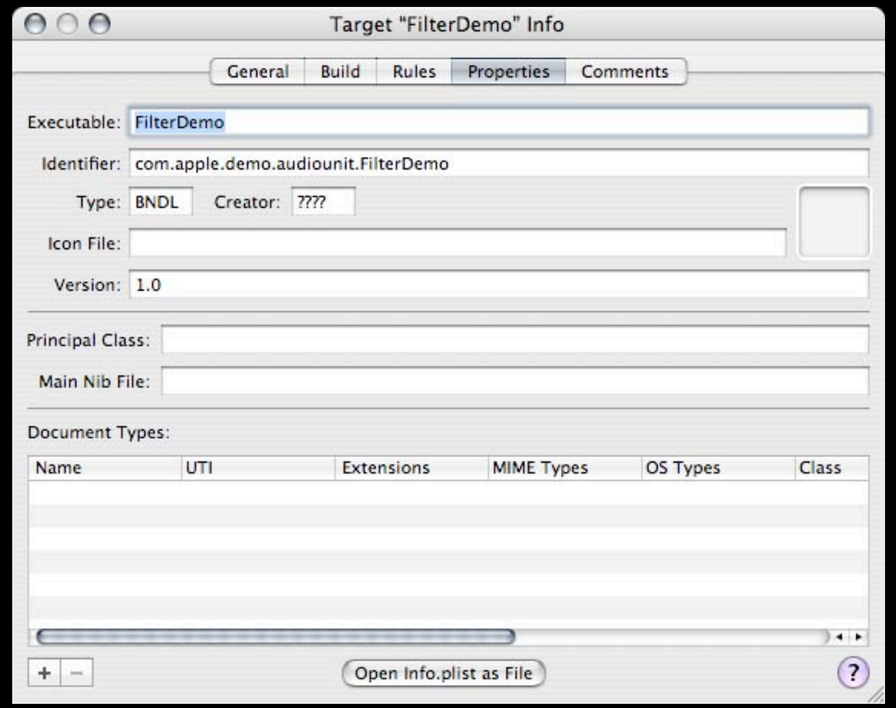
Chris Rogers
Software Engineer

http://developer.apple.com/wwdc2005/

Creating the Xcode Project

Using the Xcode Audio Unit Template

- Configure the component resources
  - Verify the .exp file
  - Set the component subtype, manufacturer, and name
- Configure the Info.plist file



Building for Intel?

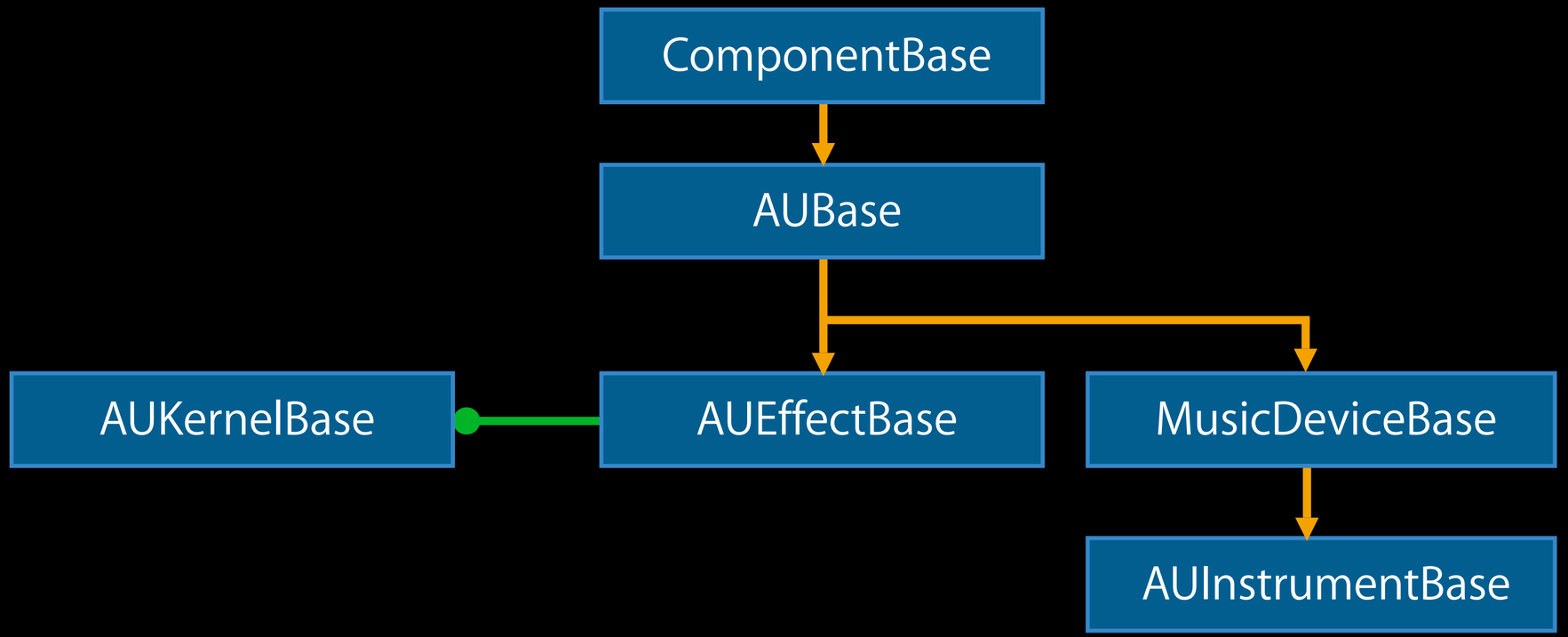
- Use the SDK
- Base classes and support files do all the work for you

Overview

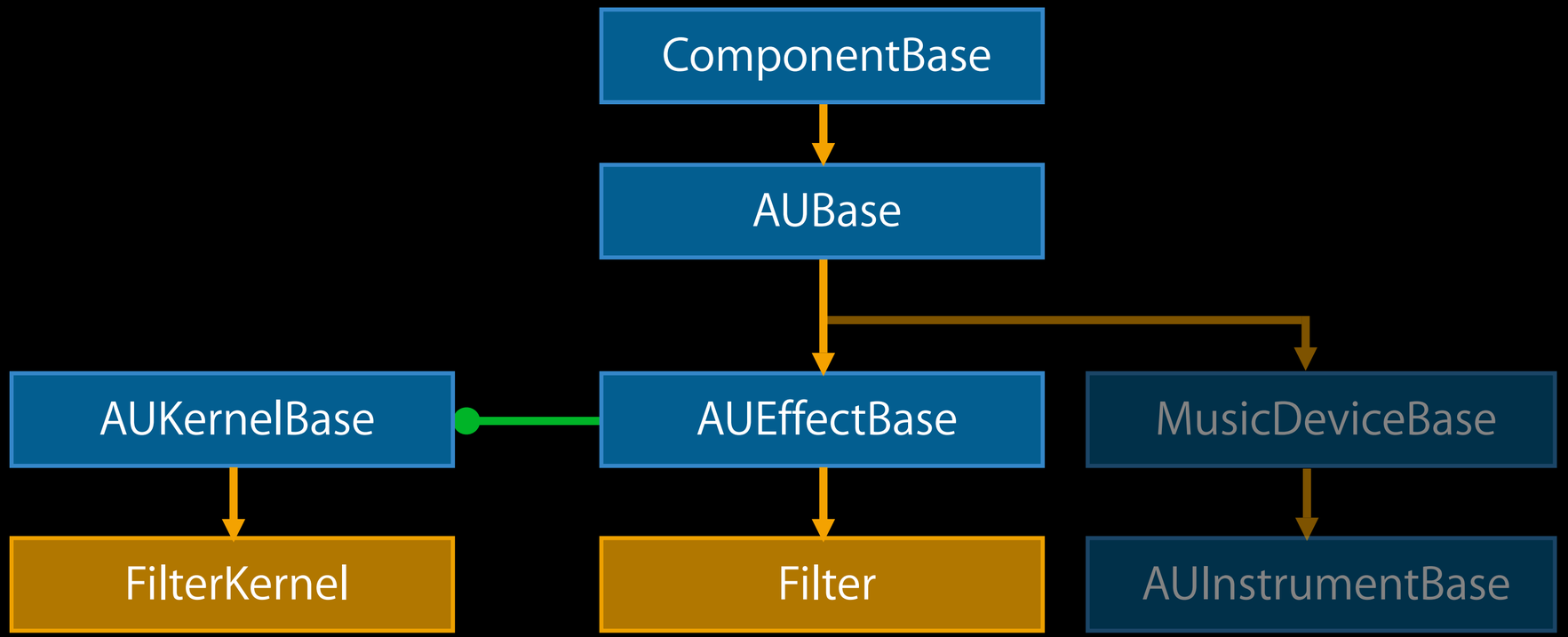
Writing an effect Audio Unit

- Base classes implement basic Audio Unit functionality
- Developer implements custom code
  - Parameters
  - Properties (optional)
  - Signal Processing

SDK Class Hierarchy



SDK Class Hierarchy



SDK Base Classes for Effects

- Subclass AUEffectBase
  - Assumes *n* → *n* independent channel processing, uses *n* AUKernelBase objects
  - Override methods for
    - Parameters
    - Properties
    - Factory Presets
- Subclass AUKernelBase
  - Put your signal processing code here

Parameters

- Provide real-time control of processing
- Examples
  - Filter cutoff frequency
  - Delay time
  - Chorus depth
- Each parameter has
  - Name
  - Unit type (Hertz, Decibels, Seconds…)
  - Min/max/default value
  - Flags

Defining Parameters

- Override GetParameterInfo() in AUEffectBase subclass
- For each parameter define
  - Name
  - Unit type
  - Min/max/default value
  - Flags
- In AUEffectBase subclass constructor, call SetParameter() to define each parameter

Properties

- General mechanism for configuring and retrieving information from an Audio Unit
- Required properties must be implemented (handled by base classes for you)
- Custom properties may be defined for passing arbitrary information between Audio Unit and its custom UI

Defining Properties

- In AUEffectBase subclass, override
  - GetPropertyInfo()
  - GetProperty()

Host Controls Audio Unit's Rendering

- Host calls AudioUnitRender()
  - Host provides
    - Context (device/offline)
    - Number of sample frames
    - Bus/time stamp
    - Audio buffer list to render into
  - Audio Unit
    - Gets input
    - Processes/writes to output audio buffer list
- Processing generates a continuous audio stream

Host Controls Audio Unit's Rendering

- Host calls AudioUnitReset() as the usage context requires
  - Timeline change
  - Mute/unmute AU's track

Writing the Signal Processing Code

- Override AUKernelBase methods
  - Process()
    - Does the interesting signal processing
    - Operates on non-interleaved source and destination buffers
  - Reset() resets filter state, clears delay buffers, etc.

Factory Presets

- List of built-in sets of parameter values
- Each factory preset consists of
  - Preset number
  - Name

Factory Presets

- Internally an Audio Unit keeps track of the parameter values associated with each preset
- Override GetPresets() to publish a list (CFArrayRef of AUPreset structs)
- Override NewFactoryPresetSet() to handle a preset change—this requires changing parameter values to match the preset

Demo

Building and validating filter demo Audio Unit

Chris Rogers
Software Engineer

http://developer.apple.com/wwdc2005/

Building a Custom Audio Unit Interface

Michael Hopkins
Software Engineer

http://developer.apple.com/wwdc2005/

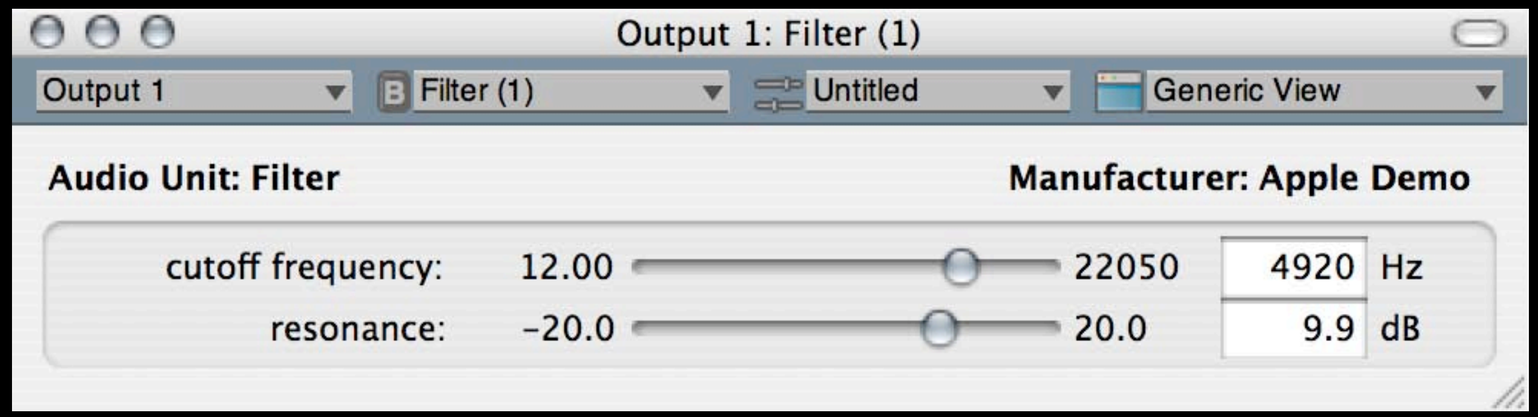
Introduction to Custom User Interfaces

Many advantages over generic view

- Hides unwanted or extraneous detail
- Graphical representation of data
- Allows flexibility and control over presentation
- Provides “branding” opportunity

Which Provides the Better User Experience?

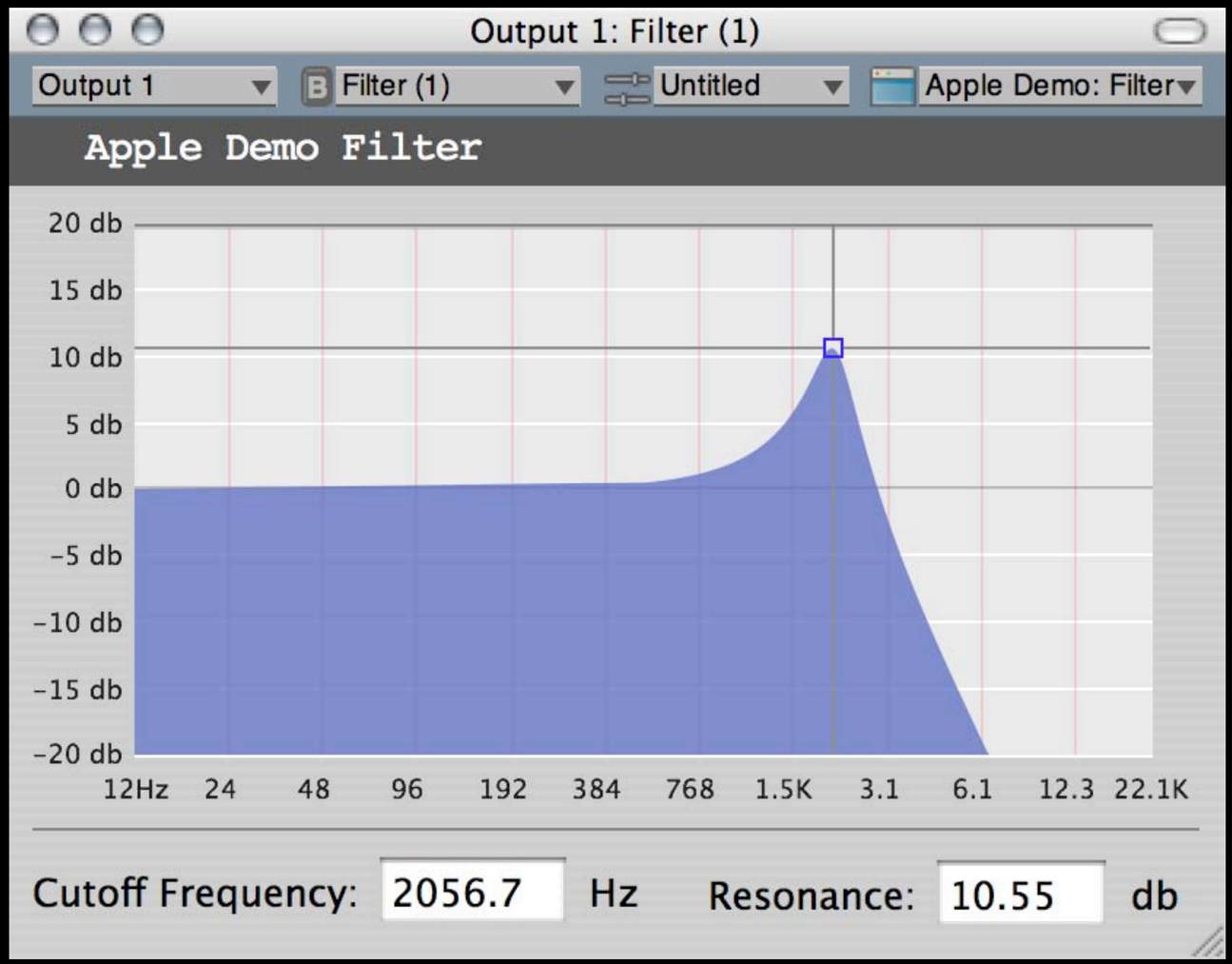
This…



Generic View

Which Provides the Better User Experience?

Or this?



Custom View

Demo

Previewing the Filter UI running in AU Lab

Michael Hopkins
Software Engineer

View Basics

- Carbon or Cocoa
  - Interface may be dynamic or nib-based
- Retrieved from the Audio Unit via property call
  - Carbon: kAudioUnitProperty_GetUIComponentList
  - Cocoa: kAudioUnitProperty_CocoaUI

General Context

- Host provides window
  - Carbon: AU View is a ControlRef
  - Cocoa: AU View is a NSView
- Drawing
- Handling Events
  - Carbon: Carbon Event Manager
  - Cocoa: NSResponder

Anatomy of an AU With a Cocoa UI

Audio Unit contained in .component file

FilterDemo.component	
Info.plist	(Bundle ID, version number, …)
MacOS	
FilterDemo	(Audio Unit executable code)
Resources	
CocoaFilterView.bundle	(Cocoa View bundle)
English.lproj	(AU localized resources)
FilterDemo.rsrc	(ID, type, subtype, manufacturer, etc)

Anatomy of an AU With a Cocoa UI

Cocoa UI View contained in Cocoa Bundle

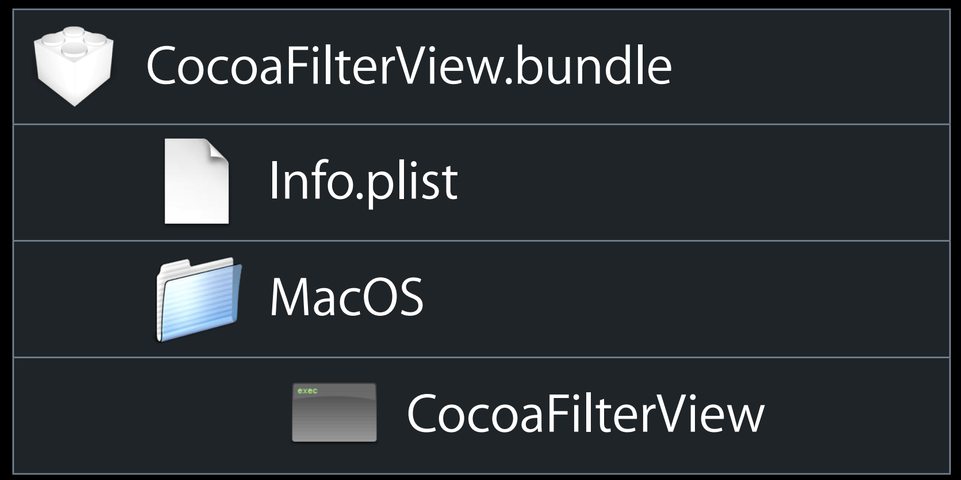
CocoaFilterView.bundle	
Info.plist	(Main Class, nib, bundle name)
MacOS	
CocoaFilterView	(Cocoa View executable code)
Resources	
English.lproj	(View localized resources)
CocoaView.nib	(Interface Builder nib file)
BackgroundImage.tiff	

Loading a Cocoa UI View

Host calls GetProperty() with kAudioUnitProperty_CocoaUI

- Returns a struct specifying location of bundle and main class name(s)

```
typedef structAudioUnitCocoaViewInfo {
  CFURLRef    mCocoaAUViewBundleLocation;
  CFStringRef mCocoaAUViewClass[1];
}
```

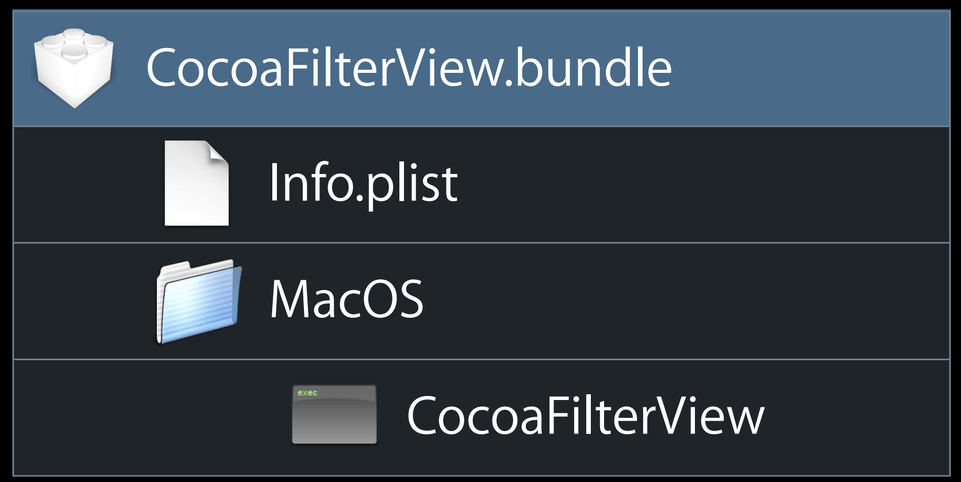


Loading a Cocoa UI View

Host calls GetProperty() with kAudioUnitProperty_CocoaUI

- Returns a struct specifying location of bundle and main class name(s)

```
typedef structAudioUnitCocoaViewInfo {
  CFURLRef    mCocoaAUViewBundleLocation;
  CFStringRef mCocoaAUViewClass[1];
}
```

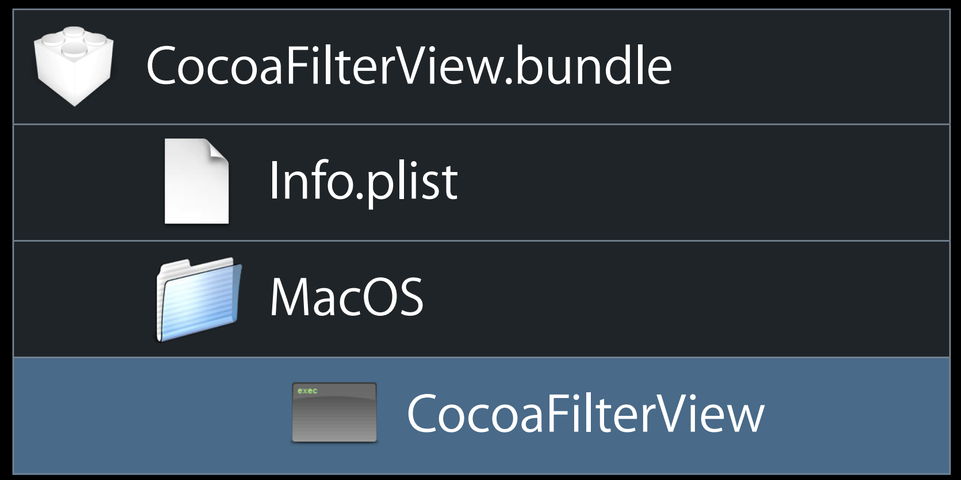


Loading a Cocoa UI View

Host calls GetProperty() with kAudioUnitProperty_CocoaUI

- Returns a struct specifying location of bundle and main class name(s)

```
typedef structAudioUnitCocoaViewInfo {
  CFURLRef    mCocoaAUViewBundleLocation;
  CFStringRef mCocoaAUViewClass[1];
}
```



Supporting kAudioUnitProperty_CocoaUI

Implementing GetPropertyInfo()

```
ComponentResult Filter::GetPropertyInfo( AudioUnitPropertyID inID,
                                         AudioUnitScope      inScope,
                                         AudioUnitElement    inElement,
                                         UInt32 &            outDataSize,
                                         Boolean &           outWritable)
{
   if (inScope == kAudioUnitScope_Global) {
      switch (inID) {
         case kAudioUnitProperty_CocoaUI
            outWritable = false;
            outDataSize = sizeof (AudioUnitCocoaViewInfo);
            return noErr;
      }
   }
}
```

Filter.cpp

Supporting kAudioUnitProperty_CocoaUI

Implementing GetProperty()

```
ComponentResult Filter::GetProperty( AudioUnitPropertyID inID,
                                     AudioUnitScope      inScope,
                                     AudioUnitElement    inElement,
                                     void *              outData)
{
   if (inScope == kAudioUnitScope_Global) {
      switch (inID) {
         case kAudioUnitProperty_CocoaUI:
            CFBundleRef bundle = CFBundleGetBundleWithIdentifier(CFSTR
                                    (“com.apple.demo.audiounit.FilterDemo”));
            if (bundle == NULL) return fnfErr;

            CFURLRef bundleURL = CFBundleCopyResourceURL(bundle,
               CFSTR(“CocoaFilterView”), CFSTR(“bundle”), NULL);
...
```

Filter.cpp

Supporting kAudioUnitProperty_CocoaUI

Implementing GetProperty() continued

```
...
case kAudioUnitProperty_CocoaUI:
  CFBundleRef bundle = CFBundleGetBundleWithIdentifier(CFSTR
                            (“com.apple.demo.audiounit.FilterDemo”));
  if (bundle == NULL) return fnfErr;

  CFURLRef bundleURL = CFBundleCopyResourceURL(bundle,
                              CFSTR(“CocoaFilterView”), CFSTR(“bundle”), NULL);
  if (bundleURL == NULL) return fnfErr;

  CFStringRef className = CFSTR(“AppleDemoFilter_ViewFactory”);
  AudioUnitCocoaViewInfo cocoaInfo = {bundleURL, className};
  *((AudioUnitCocoaViewInfo *)outData = cocoaInfo;

return noErr;
```

Filter.cpp

Supporting kAudioUnitProperty_CocoaUI

Implementing GetProperty() continued

```
...
case kAudioUnitProperty_CocoaUI:
   CFBundleRef bundle = CFBundleGetBundleWithIdentifier(CFSTR
                              (“com.apple.demo.audiounit.FilterDemo”));
   if (bundle == NULL) return fnfErr;

   CFURLRef bundleURL = CFBundleCopyResourceURL(bundle,
                              CFSTR(“CocoaFilterView”), CFSTR(“bundle”), NULL);
   if (bundleURL == NULL) return fnfErr;

   CFStringRef className = CFSTR(“AppleDemoFilter_ViewFactory”);
   AudioUnitCocoaViewInfo cocoaInfo = {bundleURL, className};
   *((AudioUnitCocoaViewInfo *)outData = cocoaInfo;

return noErr;
```

Filter.cpp

Examining the Main Class

Cocoa bundle's principal class is main entry point

- Implements AUCocoaUIBase
- Factory returning NSView sub-class via protocol method

```
-(NSView *) uiViewForAudioUnit :(AudioUnit)inAU
                      withSize :(NSSize)inPreferredSize;
```

- Owner of the nib file (if applicable)

AppleDemoFilter_ViewFactory.m

Audio Unit/View Communication

- View and AU are separate entities
  - Encapsulation boundaries should be respected
    - Don’t share pointers
- Communicating state with Audio Unit
  - Parameters
    - AudioUnitGetParameter()
    - AudioUnitSetParameter()
  - Properties
    - AudioUnitGetPropertyInfo()
    - AudioUnitGetProperty()
    - AudioUnitSetProperty()

AU Event Listener

- Notification mechanism
  - Parameter changes
  - Property changes
- Thread-safe
- Control granularity of notifications
- Used by views and host applications

Creating and Disposing Listeners

- View "listens" to Audio Unit by creating a listener
  - AUEventListenerCreate()
- Disposing listeners
  - AUEventListenerDispose()

Event Listeners

Specify interest in events

- AUEventListenerAddEventType()
- AUEventListenerRemoveEventType()

Listener Notification

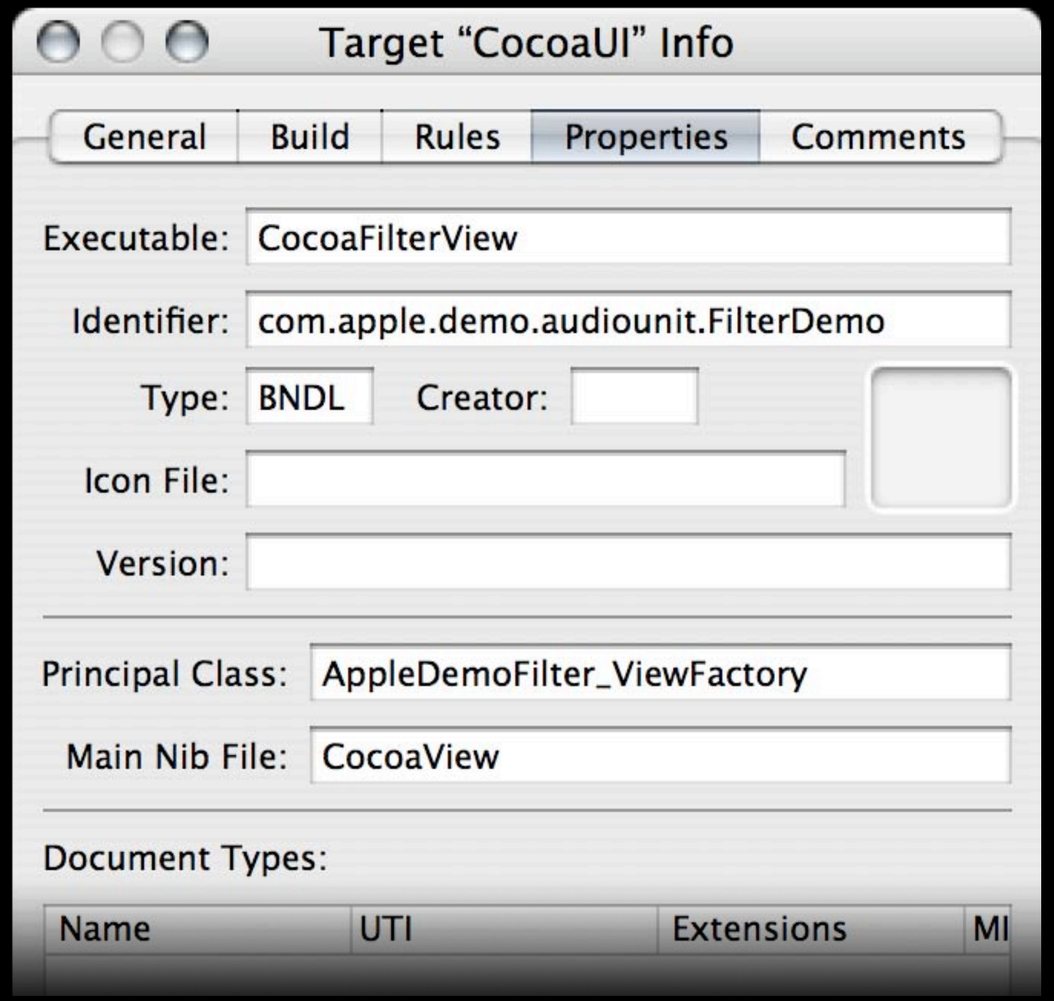
Notify listeners of parameter or event changes

- AUEventListenerNotify()
- AUParameterSet()
  - AudioUnitSetParameter()
  - AUEventListenerNotify()

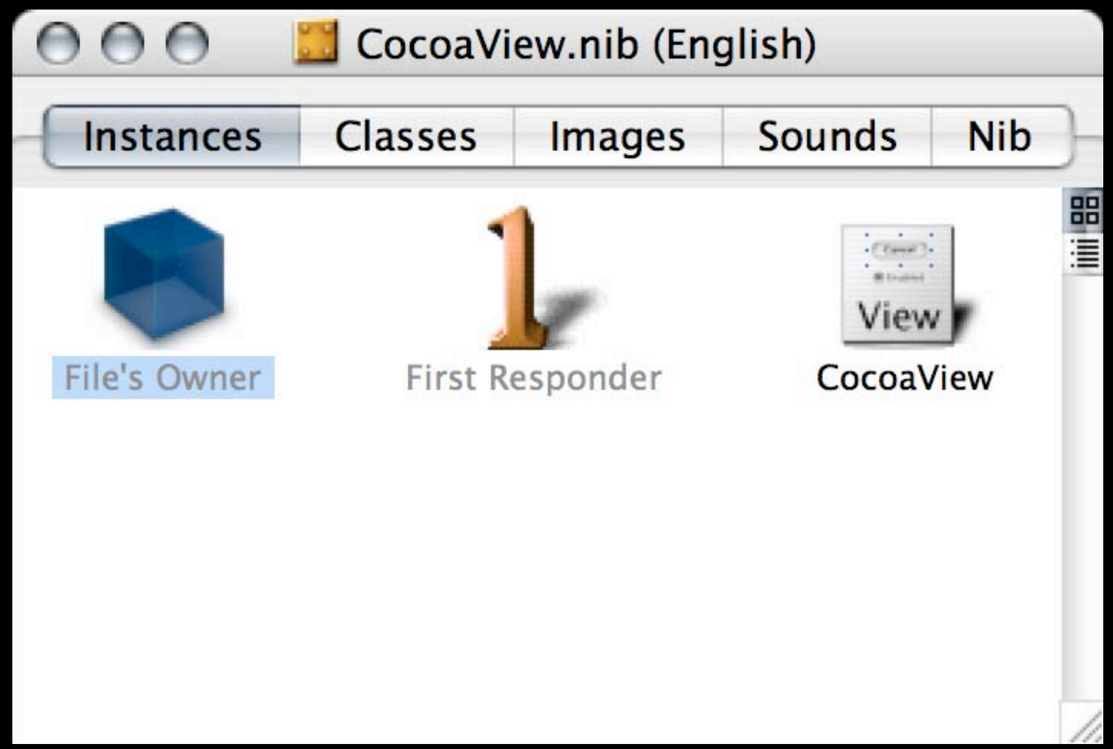
Preparing the Project

- Rename classes to prevent namespace collisions
- Update CocoaUI Xcode target
  - Bundle name
  - Identifier
  - Nib file name
  - Principal (main) class name
- Add additional resources

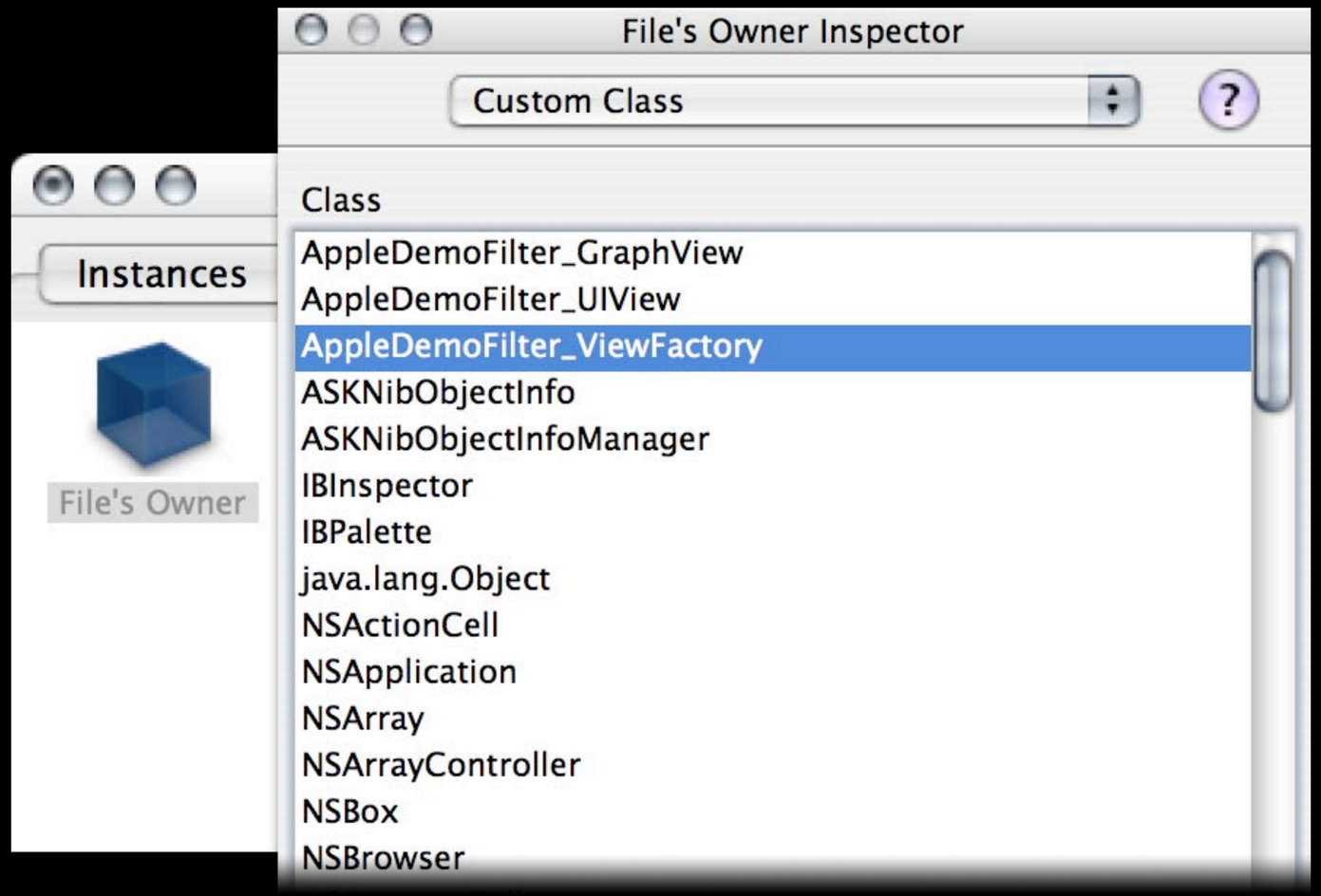
Updating the CocoaUI Target in Xcode



Updating Class Names in Interface Builder

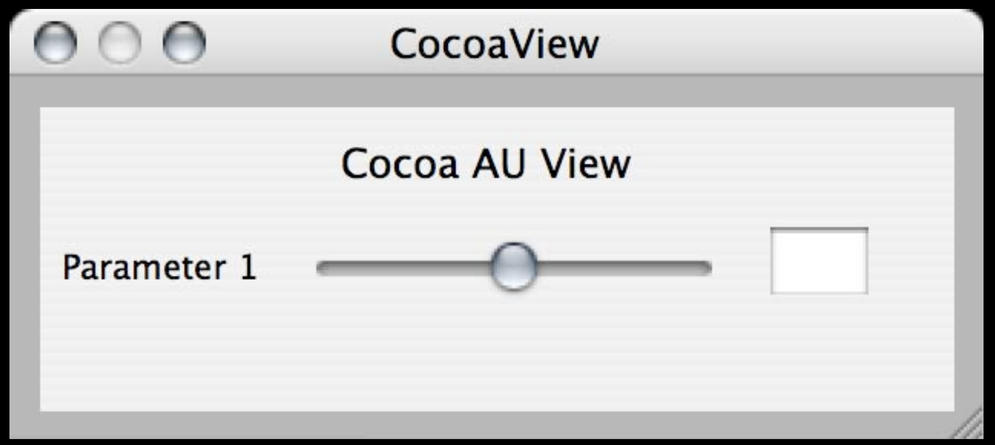


Updating Class Names in Interface Builder



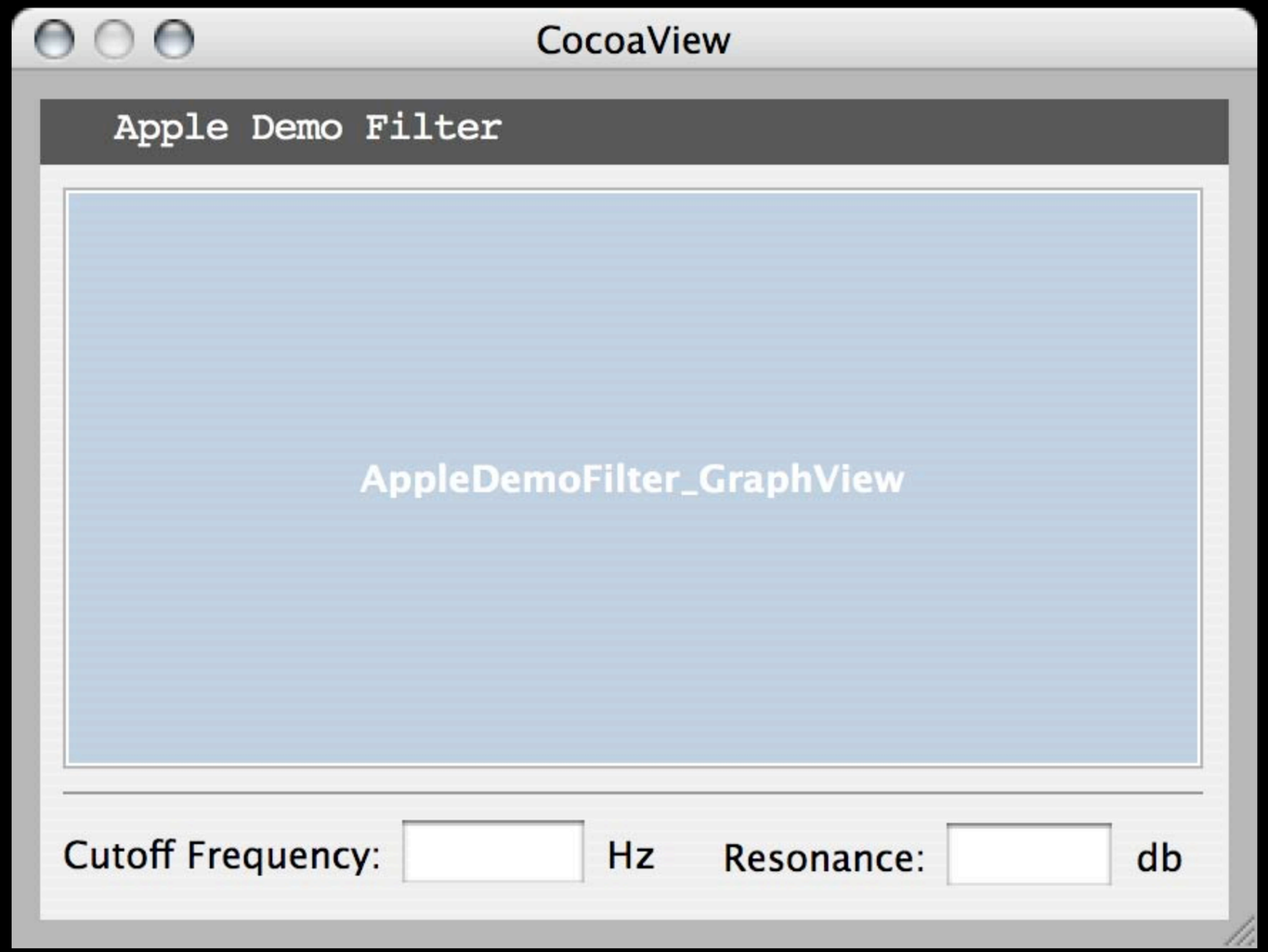
Creating the UI in Interface Builder

Remove project template default interface



Creating the UI in Interface Builder

Replace with new interface elements



View Initialization

Host

`AudioUnitGetProperty(kAudioUnitProperty_CocoaUI,…)`

Load Bundle and instantiate main class

AppleDemoFilter_ViewFactory

```
-(NSView *)uiViewForAudioUnit:(AudioUnit)inAU
                     withSize:(NSSize)inSize
```

Load nib file, create view

AppleDemoFilter_UIView

`-(void)setAU:(AudioUnit)inAU`

Initialize view, add listeners, cache au

View Initialization

Host

`AudioUnitGetProperty(kAudioUnitProperty_CocoaUI,…)`

Load Bundle and instantiate main class

AppleDemoFilter_ViewFactory

```
-(NSView *)uiViewForAudioUnit:(AudioUnit)inAU
                     withSize:(NSSize)inSize
```

Load nib file, create view

AppleDemoFilter_UIView

`-(void)setAU:(AudioUnit)inAU`

Initialize view, add listeners, cache au

View Initialization

Host

`AudioUnitGetProperty(kAudioUnitProperty_CocoaUI,…)`

Load Bundle and instantiate main class

AppleDemoFilter_ViewFactory

```
-(NSView *)uiViewForAudioUnit:(AudioUnit)inAU
                     withSize:(NSSize)inSize
```

Load nib file, create view

AppleDemoFilter_UIView

`-(void)setAU:(AudioUnit)inAU`

Initialize view, add listeners, cache au

Responding to Audio Unit Changes

Creating the event listener

```
-(void) priv_addListeners {
  if (mAU) {
     AUEventListenerCreate (EventListenerDispatcher, self,
                CFRunLoopGetCurrent(), kCFRunLoopDefaultMode,
                0.05,0.05,/* 50 ms interval, granularity */
                &mAUEventListener );

     AudioUnitEvent auEvent;
     AudioUnitParameter param = {mAU, kFilterParam_CutoffFrequency,
                                 kAudioUnitScope_Global, 0};
...
```

AppleDemoFilter_UIView.m

Responding to Audio Unit Changes

Adding cutoff frequency beginGesture listener

```
...
    AudioUnitEvent auEvent;
    AudioUnitParameter param = {mAU, kFilterParam_CutoffFrequency,
                                kAudioUnitScope_Global, 0};
    auEvent.mArgument.mParameter = parameter;
    auEvent.mEventType = kAudioUnit_BeginParameterChangeGesture;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);

    auEvent.mEventType = kAudioUnit_EndParameterChangeGesture;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);

    auEvent.mEventType = kAudioUnit_ParameterValueChanged;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);
  }
}
```

AppleDemoFilter_UIView.m

Responding to Audio Unit Changes

Adding cutoff frequency beginGesture listener

```
...
    AudioUnitEvent auEvent;
    AudioUnitParameter param = {mAU, kFilterParam_CutoffFrequency,
                                kAudioUnitScope_Global, 0};
    auEvent.mArgument.mParameter = parameter;
    auEvent.mEventType = kAudioUnit_BeginParameterChangeGesture;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);

    auEvent.mEventType = kAudioUnit_EndParameterChangeGesture;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);

    auEvent.mEventType = kAudioUnit_ParameterValueChanged;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);
  }
}
```

AppleDemoFilter_UIView.m

Responding to Audio Unit Changes

Adding cutoff frequency endGesture listener

```
...
    AudioUnitEvent auEvent;
    AudioUnitParameter param = {mAU, kFilterParam_CutoffFrequency,
                                kAudioUnitScope_Global, 0};
    auEvent.mArgument.mParameter = parameter;
    auEvent.mEventType = kAudioUnit_BeginParameterChangeGesture;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);

    auEvent.mEventType = kAudioUnit_EndParameterChangeGesture;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);

    auEvent.mEventType = kAudioUnit_ParameterValueChanged;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);
  }
}
```

AppleDemoFilter_UIView.m

Responding to Audio Unit Changes

Adding cutoff frequency valueChanged listener

```
...
    AudioUnitEvent auEvent;
    AudioUnitParameter param = {mAU, kFilterParam_CutoffFrequency,
                                kAudioUnitScope_Global, 0};
    auEvent.mArgument.mParameter = parameter;
    auEvent.mEventType = kAudioUnit_BeginParameterChangeGesture;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);

    auEvent.mEventType = kAudioUnit_EndParameterChangeGesture;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);

    auEvent.mEventType = kAudioUnit_ParameterValueChanged;
    AUEventListenerAddEventType(mAUEventListener, self, &auEvent);
  }
}
```

AppleDemoFilter_UIView.m

Responding to Audio Unit Changes

Implementing event listener dispatch routine

```
-(void) EventListenerDispatcher (void *inRefCon,
                                 void *inObject,
                                 const AudioUnitEvent *inEvent,
                                 Float32 inValue) {
  AppleDemoFilter_UIView *SELF = (AppleDemoFilter_UIView *)inRefCon;

  [SELF priv_eventListener :inObject
                    event :inEvent
                    value :inValue];
}
```

AppleDemoFilter_UIView.m

Responding to Audio Unit Changes

Implementing parameter listener method

```
-(void) priv_eventListener :(void *)inObject
                     event :(const AudioUnitEvent *)inEvent
                     value :(Float32)inValue {
  switch (inEvent->mEventType) {
     case kAudioUnitEvent_ParameterValueChange:
        switch (inParameter->mParameterID) {
           case kFilterParam_CutoffFrequency:
              [cutoffFrequencyField setFloatValue: inValue];
              [graphView setFreq: inValue];
              break;
        case kFilterParam_Resonance:
           [resonanceField setFloatValue: inValue];
           [graphView setRes: inValue];
           break;
     }
```

AppleDemoFilter_UIView.m

Responding to Audio Unit Changes

Implementing parameter listener method

```
-(void) priv_eventListener :(void *)inObject
                     event :(const AudioUnitEvent *)inEvent
                     value :(Float32)inValue {
  switch (inEvent->mEventType) {
     case kAudioUnitEvent_ParameterValueChange:
        switch (inParameter->mParameterID) {
           case kFilterParam_CutoffFrequency:
              [cutoffFrequencyField setFloatValue: inValue];
              [graphView setFreq: inValue];
              break;
        case kFilterParam_Resonance:
           [resonanceField setFloatValue: inValue];
           [graphView setRes: inValue];
           break;
     }
```

AppleDemoFilter_UIView.m

Responding to Audio Unit Changes

Handling begin gesture

```
-(void) priv_eventListener :(void *)inObject
                     event :(const AudioUnitEvent *)inEvent
                     value :(Float32)inValue {
  switch (inEvent->mEventType) {
     case kAudioUnitEvent_ParameterValueChange:
        ...
        break;
     case kAudioUnitEvent_BeginParameterChangeGesture:
        [graphView handleBeginGesture];
        break;
     case kAudioUnitEvent_EndParameterChangeGesture:
        [graphView handleEndGesture];
        break;
  }
}
```

AppleDemoFilter_UIView.m

Responding to Audio Unit Changes

Handling end gesture

```
-(void) priv_eventListener :(void *)inObject
                     event :(const AudioUnitEvent *)inEvent
                     value :(Float32)inValue {
  switch (inEvent->mEventType) {
     case kAudioUnitEvent_ParameterValueChange:
        ...
        break;
     case kAudioUnitEvent_BeginParameterChangeGesture:
        [graphView handleBeginGesture];
        break;
     case kAudioUnitEvent_EndParameterChangeGesture:
        [graphView handleEndGesture];
        break;
  }
}
```

AppleDemoFilter_UIView.m

Handling User Actions

User interacts with interface via

- Changing parameter text fields
  - Cutoff frequency
  - Resonance
- GraphView
  - Clicking in view
  - Dragging control point

User Actions in Text Fields

Handling changes to Cutoff Frequency

```
-(IBAction) cutoffFrequencyChanged:(id)sender {
  float floatValue = [sender floatValue]; // value of cutoff field
  AudioUnitParameter cutoff = {mAU,
                               kFilterParam_CutoffFrequency,
                               kAudioUnitScope_Global, 0};
  AUParameterSet(mParameterListener, sender,
                 &cutoff, (Float32)floatValue, 0);
}
```

AppleDemoFilter_UIView.m

Handling User Actions in the View

Clicking in view or dragging control point

```
-(void) handleGraphDataChanged:(NSNotification *) aNotification {
  float resonance = [graphView getRes];
  float cutoff = [graphView getFreq];

  AudioUnitParameter coParam = {mAU, kFilterParam_CutoffFrequency,
                                kAudioUnitScope_Global, 0};
  AudioUnitParameter reParam = {mAU, kFilterParam_Resonance,
                                kAudioUnitScope_Global, 0};

  AUParameterSet(mParameterListener, cutoffFrequencyField,
                           &coParam, (Float32)cutoff, 0);
  AUParameterSet(mParameterListener, resonanceField,
                           &reParam, (Float32)resonance, 0);
}
```

AppleDemoFilter_UIView.m

Gesture Notifications

Sending BeginGesture notifications

```
-(void) beginGesture:(NSNotification *)aNotification {
  AudioUnitEvent event;
  AudioUnitParameter param = {mAU, kFilterParam_CutoffFrequency,
                              kAudioUnitScope_Global, 0};
  event.mArgument.mParameter = parameter;
  event.mEventType = kAudioUnitEvent_BeginParameterChangeGesture;

  AUEventListenerNotify (mAUEventListener, self, &event);

  event.mArgument.mParameter.mParameterID = kFilterParam_Resonance;
  AUEventListenerNotify (mAUEventListener, self, &event);
}
```

Gesture Notifications

Sending BeginGesture notifications

```
-(void) beginGesture:(NSNotification *)aNotification {
  AudioUnitEvent event;
  AudioUnitParameter param = {mAU, kFilterParam_CutoffFrequency,
                              kAudioUnitScope_Global, 0};
  event.mArgument.mParameter = parameter;
  event.mEventType = kAudioUnitEvent_BeginParameterChangeGesture;

  AUEventListenerNotify (mAUEventListener, self, &event);

  event.mArgument.mParameter.mParameterID = kFilterParam_Resonance;
  AUEventListenerNotify (mAUEventListener, self, &event);
}
```

Gesture Notifications

Sending EndGesture notifications

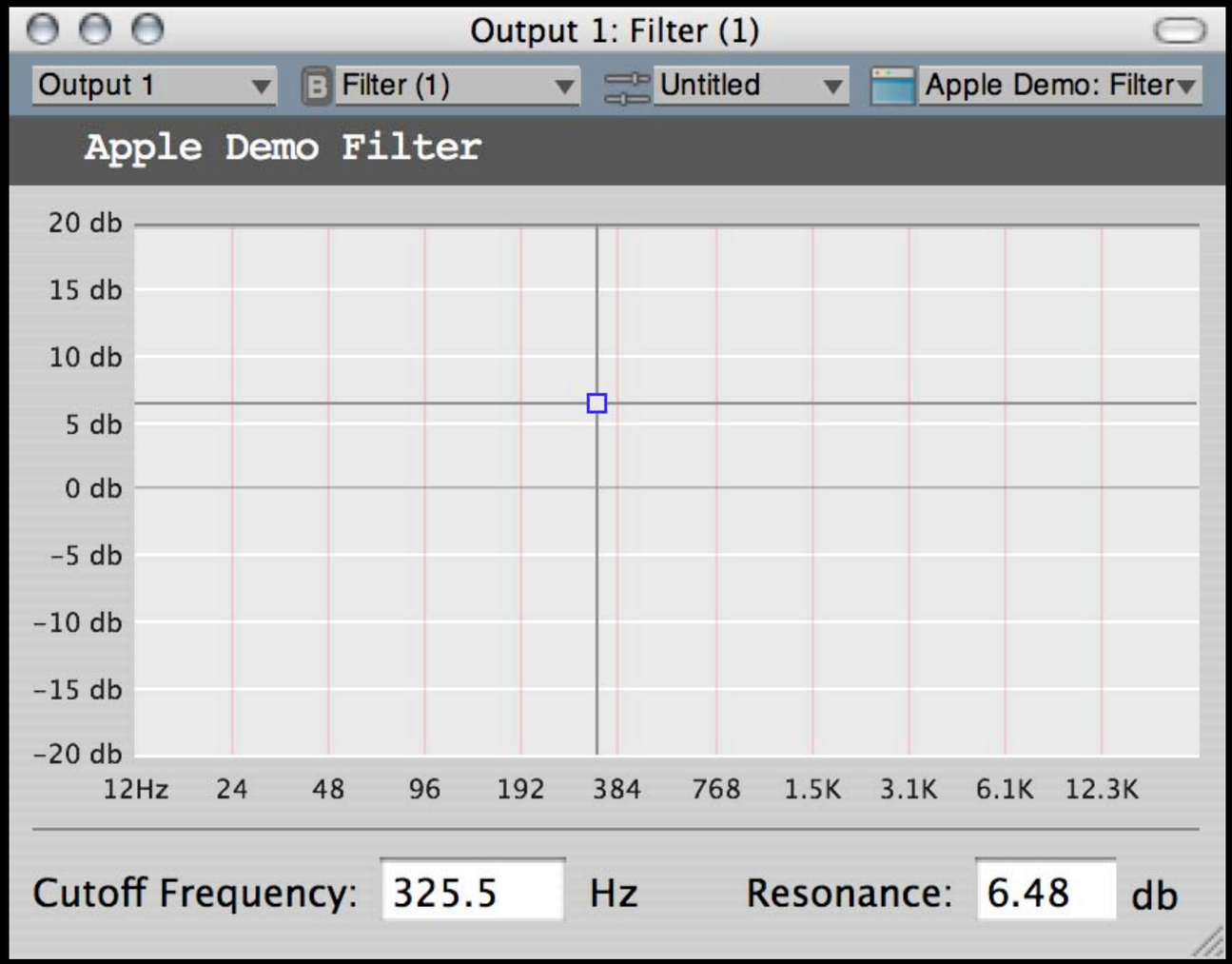
```
-(void) endGesture:(NSNotification *)aNotification {
  AudioUnitEvent event;
  AudioUnitParameter param = {mAU, kFilterParam_CutoffFrequency,
                              kAudioUnitScope_Global, 0};
  event.mArgument.mParameter = parameter;
  event.mEventType = kAudioUnitEvent_EndParameterChangeGesture;

  AUEventListenerNotify (mAUEventListener, self, &event);

  event.mArgument.mParameter.mParameterID = kFilterParam_Resonance;
  AUEventListenerNotify (mAUEventListener, self, &event);
}
```

Finishing the User Interface

- Our UI can handle
  - Parameter changes
  - Cutoff frequency text field
  - Resonance text field
  - Interaction with graph view
  - Gestures

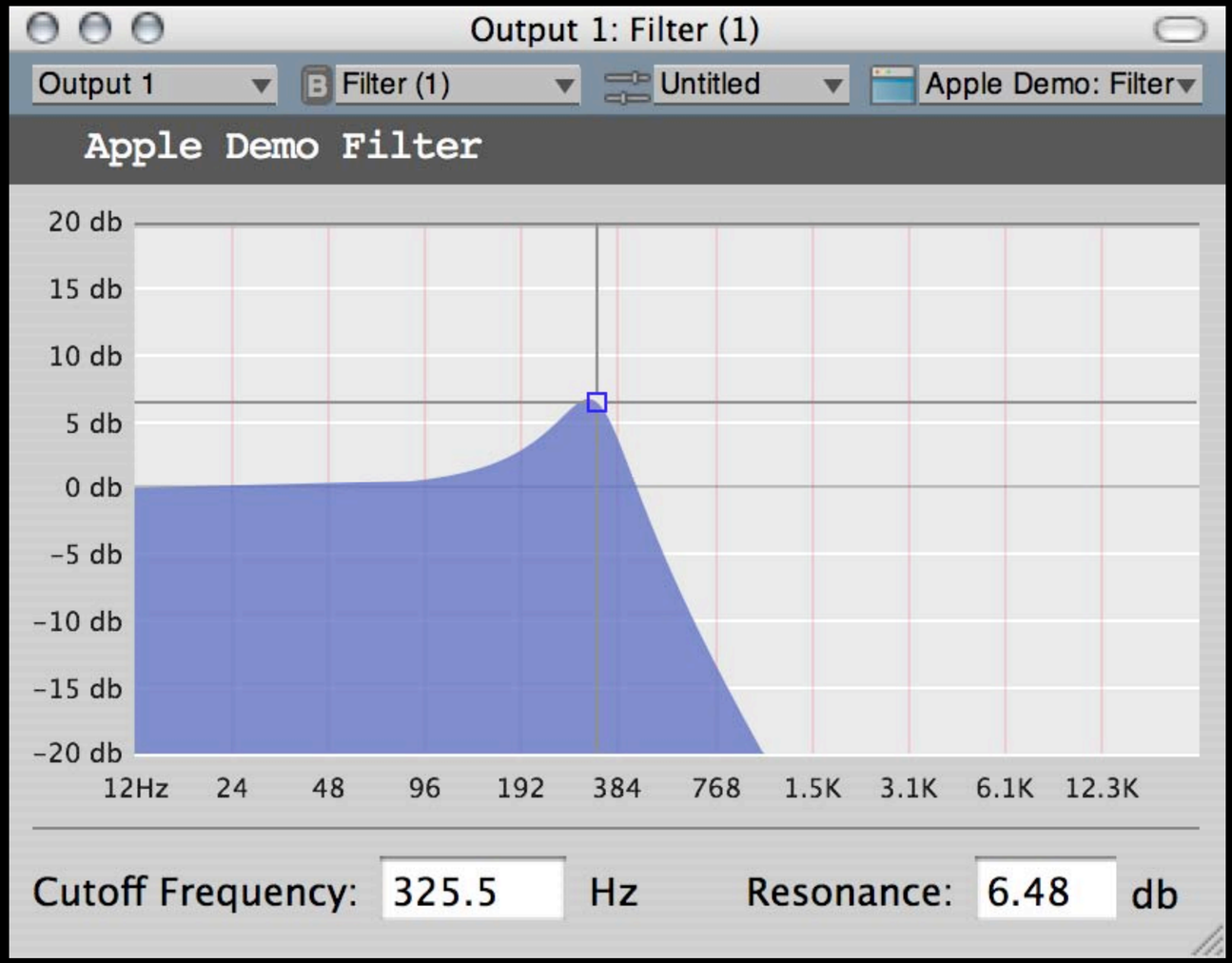
A Curve Quandary

But, how do we go from this…



A Curve Quandary

To this…?



Adding a Custom Property

- Allows Audio Unit and its view to communicate arbitrary custom information
- Information is passed with void* pointer and UInt32 length parameter

Property Methods

- Define a custom AudioUnitPropertyID (custom IDs are 64000 or higher)
- Must override methods of AUEffectBase
  - `GetPropertyInfo()`
  - `GetProperty()`
  - `SetProperty()` (if property is writable)

Demo

Adding a custom property to Filter Demo

Chris Rogers
Software Engineer

http://developer.apple.com/wwdc2005/

Drawing the Frequency Curve

Michael Hopkins
Software Engineer

http://developer.apple.com/wwdc2005/

Drawing the Frequency Response Curve

User interface must do the following

- Allocate memory for frequency response data
- Initialize data with frequency of each value at pixel location where corresponding decibel value will be plotted
- Call AudioUnitGetProperty() to get curve data from Audio Unit
- Plot the curve data

Initializing the Frequency Data

Initialized when AU is set and when window changes size

```
-(FrequencyResponse *) prepareDataForDrawing:(FrequencyResponse *) data
{
   float width = mGraphFrame.size.width;
   float rightEdge = width + mGraphFrame.origin;
   int i, pixelRatio = (int) ceil(width/kNumberOfResponseFrequencies);
   float location = mGraphFrame.origin.x;

   for (i = 0; i < kNumberOfResponseFrequencies; i++) {
      data[i].mFrequency = [self freqValueForLocation: location];
      location += pixelRatio;
   }
   return data;
}
```

AppleDemoFilter_GraphView.m

Getting the Audio Unit Property

The curve data is retrieved in our parameter listener

```
-(void) priv_eventListener   :(void *)inObject
                       event :(const AudioUnitEvent *)inEvent
                       value :(Float32)inValue {
  switch (inEvent->mEventType) {
     case kAudioUnitEvent_ParameterValueChange:
        ...
        UInt32 dataSize = kNumberOfResponseFrequencies *
                          sizeof(FrequencyResponse);
        ComponentResult result = AudioUnitGetProperty (mAU,
              kAudioUnitCustomProperty_FilterFrequencyResponse,
              kAudioUnitScope_Global, 0, mData, &dataSize);
        if (result == noErr)
           [graphView plotData: mData];
        break;
  }
}
```

AppleDemoFilter_UIView.m

Getting the Audio Unit Property

The curve data is retrieved in our parameter listener

```
-(void) priv_eventListener   :(void *)inObject
                       event :(const AudioUnitEvent *)inEvent
                       value :(Float32)inValue {
  switch (inEvent->mEventType) {
     case kAudioUnitEvent_ParameterValueChange:
        ...
        UInt32 dataSize = kNumberOfResponseFrequencies *
                          sizeof(FrequencyResponse);
        ComponentResult result = AudioUnitGetProperty (mAU,
              kAudioUnitCustomProperty_FilterFrequencyResponse,
              kAudioUnitScope_Global, 0, mData, &dataSize);
        if (result == noErr)
           [graphView plotData: mData];
        break;
  }
}
```

AppleDemoFilter_UIView.m

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Related Sessions

3D Environmental Audio with OpenAL	Session 205 Tuesday 3:30PM, Russian Hill
Lab: Core Audio, Audio Units & OpenAL	Lab Wednesday 12:00PM
ADC Plugfest	Special Event Thursday 6:00PM, Apple Campus

Who to Contact

William Stewart
Core Audio Engineering Manager
bills@apple.com

Tips

- Download and use the Xcode Audio Unit templates
- Register your manufacturer code
  http://developer.apple.com/datatype/
- Use auval command-line tool to validate Audio Unit
- Use AU Lab to test your Audio Unit

Q&A

William Stewart
Core Audio Engineering Manager
bills@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference